

# Käyttöjärjestelmät II

## MUISTINHALLINNAN OHJELMISTO Ch 7-8 [Stal 05]

# Lähtötiedot / seuraavaksi?

## **Yksinkertainen muistinhallinta (Ch 7)**

Tehtävät, staattinen vs. dynaaminen partitointi,  
Buddy System sivutus, segmentointi

## **Virtuaalimuisti (Ch 8.1)**

Sivutus ja segmentointi, sivu- ja segm. taulut, MMU

## **Laitteistotuki (Tietokoneen rakenne -kurssi)**

MMU:n sisäinen rakenne

TLB

## **Seuraavaksi KJ-II:ssa**

n **KJ:n muistinhallinta-algoritmit** (Ch 8.2)

Luento 5

n **UNIX ja Solaris: muistinhallinta** (Ch 8.3)

Luento 6

n **Linux: muistinhallinta** (Ch 8.4)

n **W2K: muistinhallinta** (Ch 8.5)

# Perusvalinnat

## n Virtuaalimuistia vai ei?

- u originaali UNIX ja MS-DOS eivät käyttäneet
  - F ei tarvittavaa laitteistotukea (MMU, TLB)

## n Sivutus vai segmentointi?

- u pelkkään segmentointiin perustuvat häviämässä
- u molempien yhteiskäyttö yleistymässä
- u nykyisin lähes aina monta tasoa

## n Millaisia algoritmeja käytetään?

- u monta tekijää vaikuttamassa
  - F reaaliaikaisuus? palvelinkone?
  - F laitteistotuki?

## n Jatkossa käsitellään vain sivutusta

# Looginen vs. fyysinen osoite

1502 → 6766

(Fig 7.12 [Stal05])

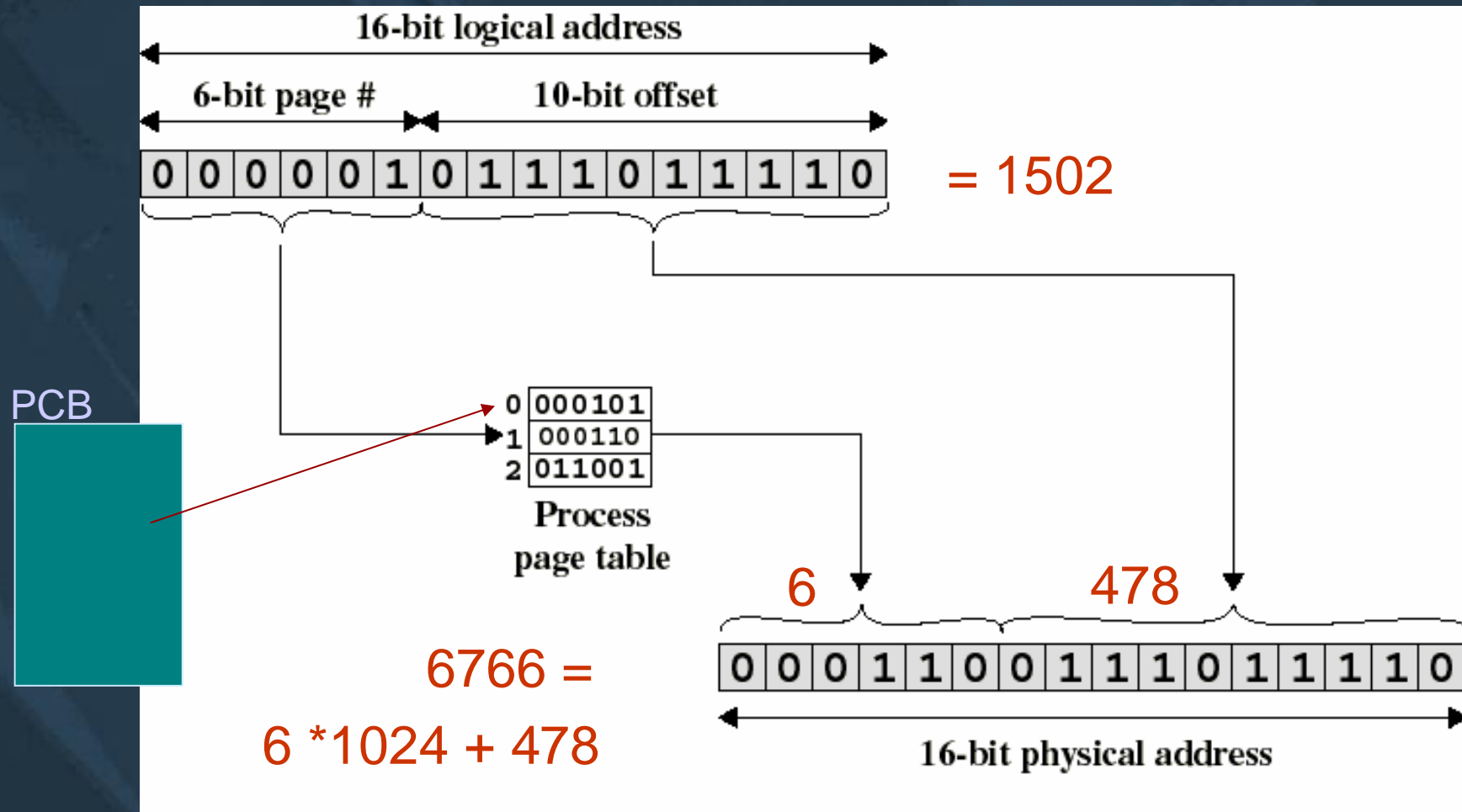
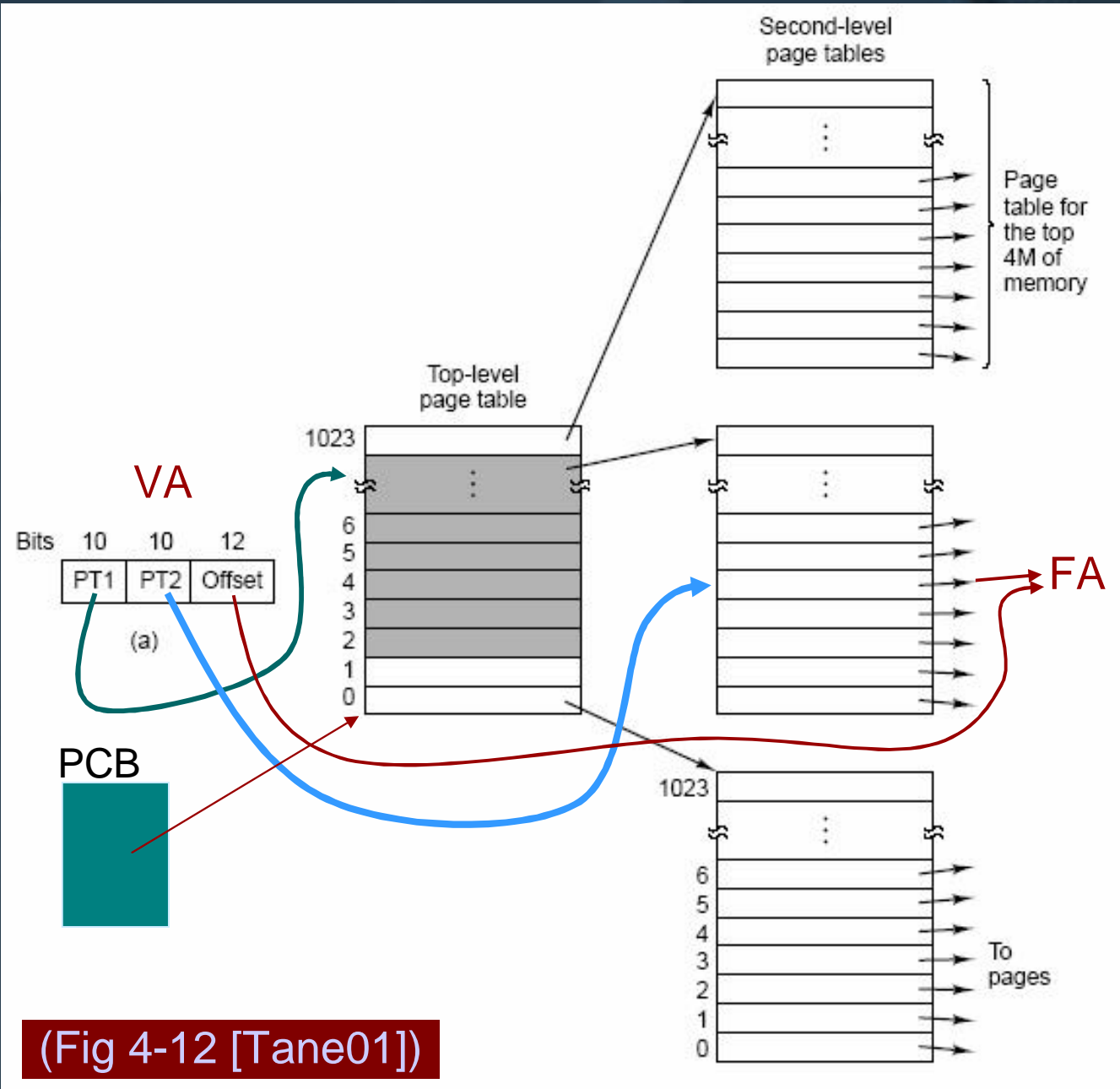


Fig 8.13 [Stal05]

# Moni- tasoinen sivutus



# Sivun koko

- n Optimoi sisäistä pirstoutumista → pieni
- n Optimoi sivutaulun (taulujen) kokoa → iso
- n Monikerta (1x, 2x, ...) levylohkon koosta
- n Optimaaliarvo erilainen eri ohjelmille
- n Optimoi TLB:n osumasuhdetta → iso
- n Sovellus tai KJ voi vaihdella sivun kokoa?

Tbl 8.2 [Stal05]

# Optimaalinen sivun koko?

n **Minimoi prosessille hukkatila** (p. 238 [Tane01])

u hukkatila = sivutaulun koko plus sisäinen pirstoutuminen?

$$f = se/p + p/2$$

s = keskim. prosessin koko tavuina  
e = sivutaulu entryn koko  
p = sivun koko  
f = hukkatila

$$df/dp = -se/p^2 + 1/2 = 0, \text{ kun } p = \sqrt{2se}$$

$$s = 1 \text{ MB } e = 8\text{B} \rightarrow p_{\text{opt}} = 4 \text{ KB}$$

$$s = 100 \text{ MB } e = 8\text{B} \rightarrow p_{\text{opt}} = 40 \text{ KB}$$

Tbl 8.2 [Stal05]

u sovellukset suurempia  $\rightarrow$  optimi sivun koko suurempi!

F kriteerinä tässä vain hukkatilan määrä

F mikä muu voisi olla kriteerinä?



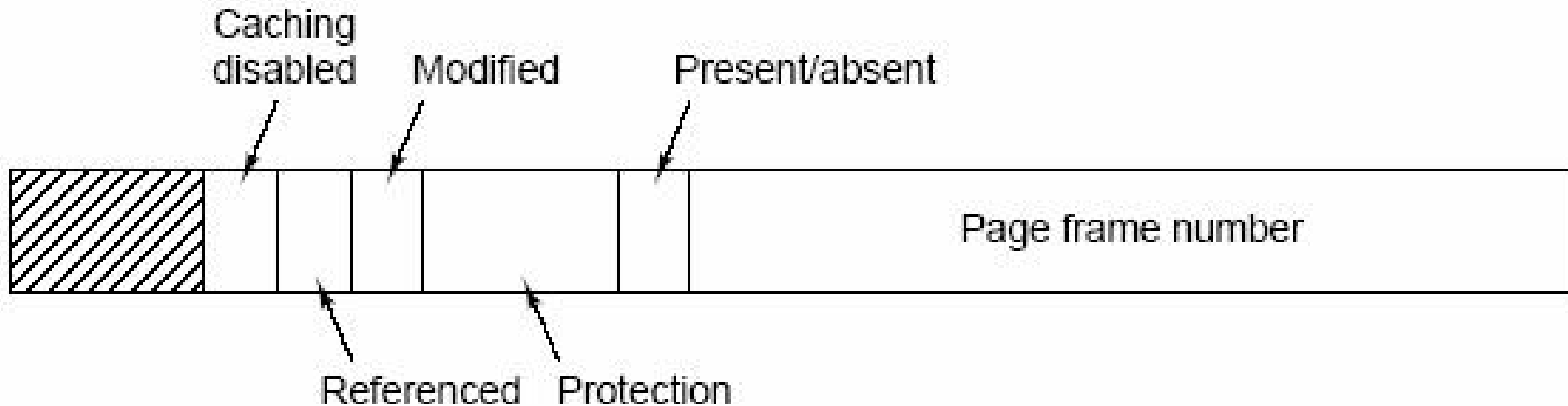
# Sivutaulun alkio

## n Jokaisella prosessilla oma sivutaulunsa

- u missä sivukehyksissä tämän prosessin sivut sijaitsevat?
- u sivutaulun fyysinen osoite PCB:ssä

port mem?  
shared mem?

dirty?



since when? R/W/E?

(Fig 4-13 [Tane01])



# TLB

- n Miten osoitteen muutos nopeasti yleensä?
- n Samalla teknologialla kuin (tason 1) välimuisti?
  - u vain ajallinen paikallisuus
- n Hudin käsittely laitteistolla tai TLB-keskeytyksellä

Fig 8.7 [Stal05]

Fig 8.10 [Stal05]

Valid	Virtual page	Modified	Protection	Page frame
1	140	1	RW	31
1	20	0	R X	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	R X	50
1	21	0	R X	45
1	860	1	RW	14
1	861	1	RW	75

Fig. 4-14. A TLB to speed up paging. [Tane01]

# TLB tarkemmin? → Tietokoneen rakenne -kurssi

Fig. 5.47 from  
Hennessy-Patterson,  
Computer Architecture

DEC Alpha AXP 21064  
memory hierarchy

Fully assoc, 12 entry  
instruction TLB

8 KB, direct mapped,  
256 line (each 32B)  
instruction cache

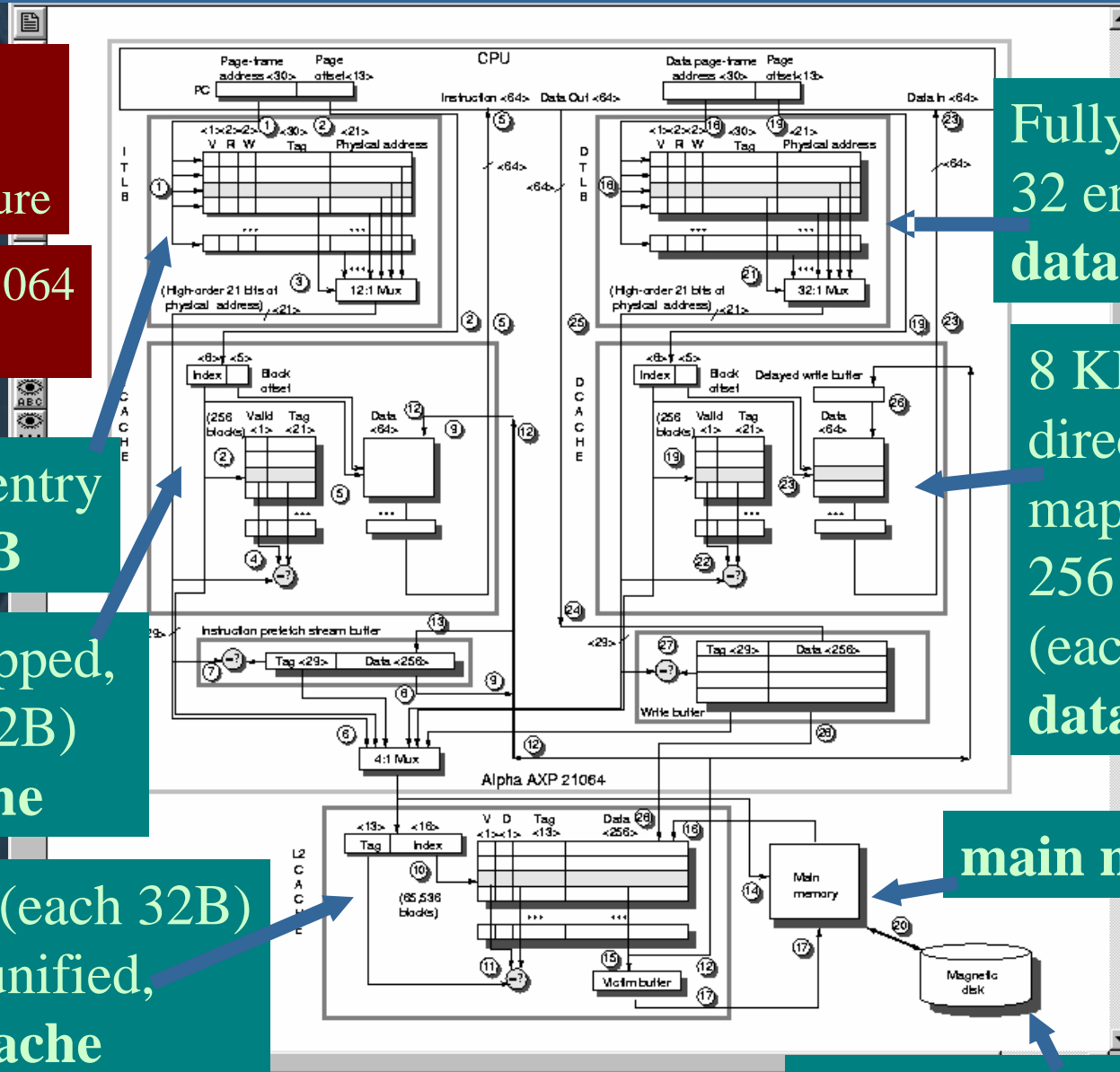
2 MB, 64K line (each 32B)  
direct mapped, unified,  
write-back L2 cache

Fully assoc,  
32 entry  
data TLB

8 KB,  
direct  
mapped,  
256 line  
(each 32B)  
data cache

main memory

paging disk (dma)



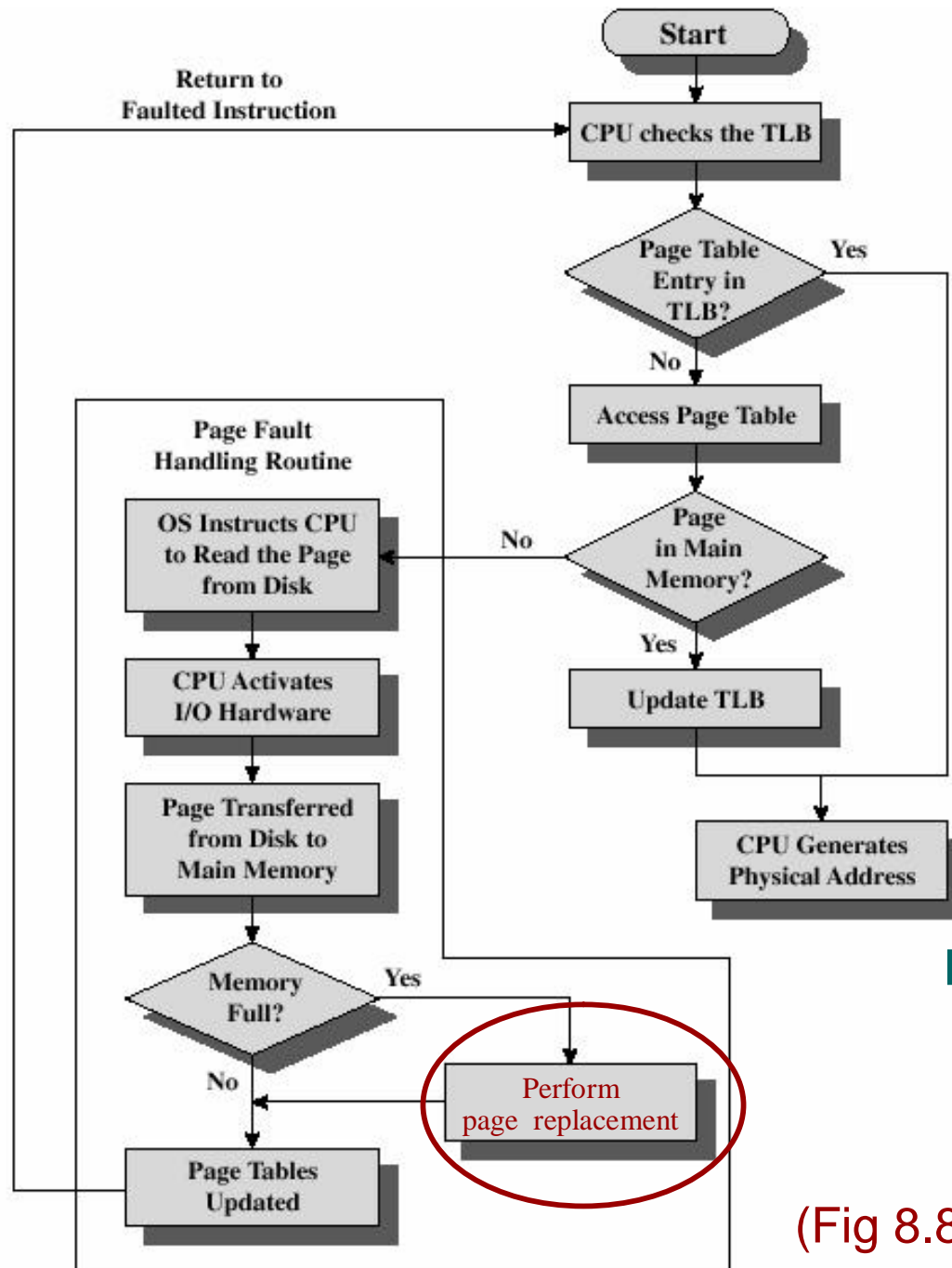
# Muistinhallinta-algoritmit

- n **Noutopolitiikka** (fetch policy)
  - u milloin sivu tuodaan muistiin?
- n **Sijoituspolitiikka** (placement policy)
  - u minne sivu sijoitetaan?
- n **Korvauspolitiikka, poistopolitiikka** (replacement policy)
  - u mikä sivu korvataan uudella ja milloin?
- n **Levyille kirjoitus -politiikka** (cleaning policy)
  - u milloin muutettu sivu vapautetaan?
- n **Moniajoaste** (load control)
  - u montako prosessia suoritettavana?
  - u paljonko muistia käytettävissä per prosessi?
- n **Käyttäjöjoukon koko** (working set)
  - u paljonko sivukehyksiä per prosessi? (resident set)
  - u mihin viitattu viime aikoina?

Tbl 8.3 [Stal05]

# Käyttöjärjestelmät II

## Noutopolitiikka



Muut osat  
käsitlety  
KJ-I:ssä

(Fig 8.8 [Stal05])

# Milloin sivu muistiin levyltä?

## n **Tarvesivutus** (demand paging)

- u tuo vasta kun viitataan sivulla olevaan osoitteeseen
- u aluksi paljon sivunpuutoksia, voi kestää kauan
  - F voi olla hyvin huono juttu, esim. puhelinkeskus
- u hetken päästä paikallisuus alkaa olla OK

## n **Ennaltanouto** (prepaging)

- u tuo etukäteen useita sivuja (tai kaikki?)
- u sivunpuutoksen sattuessa tuo monta sivua
  - F alueellinen paikallisuus
- u peräkkäisiä sivuja → optimoi hakuviiveet
- u viitataanko tuotuun sivuun?

## n **Välimuotoja**

- u tuo tärkeät ensin (demand)
- u tuo taustalla loput (prepaging)

Windows DLL epäsuora  
dynaaminen linkitys  
(implicit linking)

# Käyttöjärjestelmät II

## *Sijoituspolitiikka*



# Minne sivu sijoitetaan?

n Kun noudetaan levyltä muistiin, ...

n Segmentointi

- u Best-fit, first-fit, ... (Ch 7)

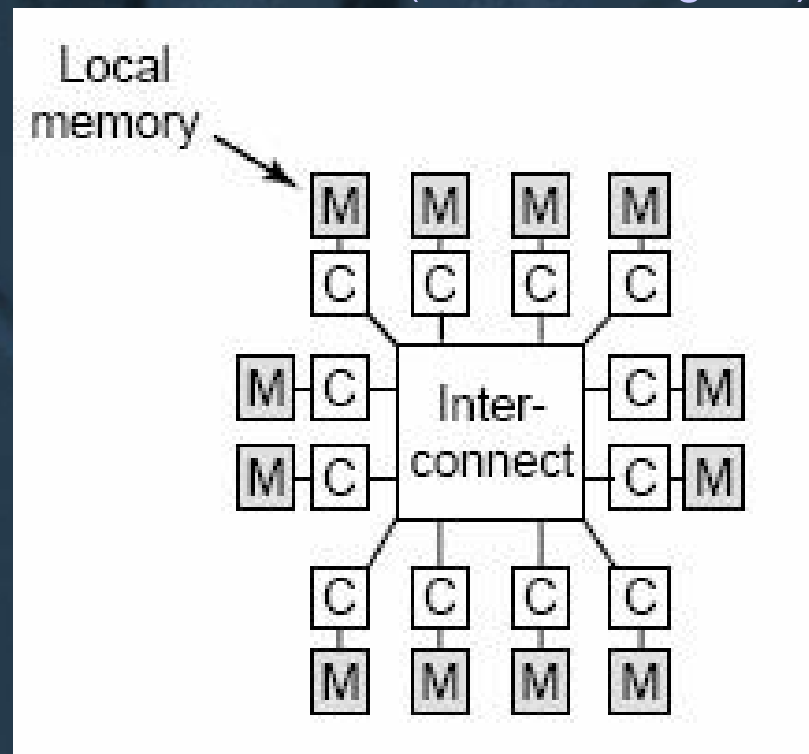
n Sivutus ja segmentoiva sivutus

- u mikä tahansa vapaa sivutila
- u osoitemuunnos MMU:ssa aina yhtä tehokkaasti

n NUMA moniprosessorit

- u nonuniform memory access
- u yhteiskäytössä oleva muisti
- u tiedon fyysinen sijainti vaikuttaa muistiviitteen nopeuteen
- u sijoituspolitiikasta taas tärkeä?
- u sivu viitteen tekijälle vai omistajalle?

(Tane01, Fig 8-1)



# Käyttöjärjestelmät II

## Korvauspolitiikka

# Muistitilan hallinta (resident set mgt)

- n Montako sivutilaa maksimi per prosessi eli montako sivua pidetään muistissa? (resident set size)
  - u kiinteä lkm?
    - F miten iso? sama kaikilla?
  - u vaihteleva lkm?
    - F millä perusteella lkm vaihtelee?
    - F käyttäjökoko (working set): “niiden sivujen joukko, jotka tarvitaan lähitulevaisuudessa”
      - resident set: “ne sivut, jotka ovat nyt muistissa”
- n Mistä kaikkialta korvattavaa sivua etsitään?
  - u etsi kaikkien sivujen joukosta (globaali politiikka)
  - u etsi prosessin omien sivujen joukosta (lokaali politiikka)
    - F voi olla silti vaihteleva sivukehysten lkm

# Sivun korvaus

## n Milloin korvataan?

- u sivunpuutos
- u muisti liian täynnä, ei vapaata tilaa (yleensä on!)
- u ei tarpeeksi vapaata tilaa

## n Mikä sivu korvataan?

- u sellainen, jota ei tarvita (lähi)tulevaisuudessa
  - F yritä ennustaa menneisyyden perusteella
- u paikallisuus: jos sivuun ei enää viitata, on ohjelma ohittanut ko. vaiheen
  - F menneisyys ei ennustanutkaan tulevaisuutta oikein!

Fig 8.1 [Stal05]

# Prosessille allokoitun muistin hallinta (Resident Set Management)

## n Allokoitu muistin määrä (sivukehysten lkm)

u jos liian pieni, niin

F muistiin mahtuu useampi prosessi

F enemmän sivunpuutoskeskeytyksiä

- liikaa? trashing eli ruuhkautuminen?

Fig 8.1 [Stal05]

u jos turhan iso

F vie tilaa muilta prosesseilta

F CPU utilisaatio turhan pieni

F "tuhlattu" muistia

Fig 8.11 (b) [Stal05]

# Muistiallokoinnin koko vs. poistettavan sivun valinta

	Local Replacement	Global Replacement
Fixed Allocation	<ul style="list-style-type: none"><li>•Number of frames allocated to process is fixed.</li><li>•Page to be replaced is chosen from among the frames allocated to that process.</li></ul>	<ul style="list-style-type: none"><li>•Not possible.</li></ul>
Variable Allocation	<ul style="list-style-type: none"><li>•The number of frames allocated to a process may be changed from time to time, to maintain the working set of the process. (esim. PFF, kalvo 35)</li><li>•Page to be replaced is chosen from among the frames allocated to that process.</li></ul>	<ul style="list-style-type: none"><li>•Page to be replaced is chosen from all available frames in main memory; this causes the size of the resident set of processes to vary.</li></ul>

(Tbl 8.4 [Stal05])



# Lukitus

## n Sivu voidaan lukita muistiin

- u ytimeen kuuluvat sivut
- u KJ:n keskeiset tietorakenteet
- u siirränän puskurit (siirron ajaksi)
- u reaaliaikajärjestelmissä myös prosessin sivut

## n Toteutus

- u sivutilataulussa ko. sivutilan kohdalla lukkobitti
  - F vapaat / varatut sivutilat
- u tai sivutaulun alkiossa lukkobitti
  - F tietylle prosessille varatut sivutilat
- u globaali vs. lokaali politiikka



# Korvauspolitiikka

- n **Optimaalinen algoritmi (OPT)** korvaa sivun, johon ei tule enää viittauksia, tai sivun, johon uudelleenviittaamiseen menee pisin aika
  - u mistäpäsen sen tiedät?
- n **Perusalgoritmit**
  - u LRU, Least recently used
    - F poista se, johon viitattu kauimmin aikaa sitten
  - u FIFO, First-in-first-out
    - F poista se, joka ollut kauimmin muistissa
  - u Clock
    - F poista se, jota ei käytetty edellisen tutkinnan jälkeen
    - F 'rengasalgoritmi'
- n **Hyvyyden mitta?**
  - u sivunpuutoskeskeytysten lukumäärä (tiheys)

on muitakin...

# OPT-poistoalgoritmi

- n **Poista se, jonka seuraavaan viittaukseen on eniten aikaa**
  - u vaatii erinomaisia ennustajan taitoja
  - u voidaan mitata jälkeenpäin trace-tiedostosta
- n **Jälkiviisaus on hienoa, mutta mitä sillä tekee?**
- n **Vertailukohta poistoalgoritmeja arvioitaessa**
  - u jos päästään lähelle OPTia, niin ollaan jo aika hyviä
  - u jos ollaan kaukana OPTista, niin huonosti menee

Fig 8.15 [Stal05]

# LRU poistoalgoritmi

- n Korvaa sivu, johon viittaamisesta kulunut kauimmin menneisyydessä
  - u vrt OPT: ... *kauimmin tulevaisuudessa*
- n Paikallisuus vihjaa tähän suuntaan
- n Toteutuksen vaikeus
  - u jokaiseen sivuun mukaan aikaleimako?
  - u miten aikaleima päivitetään?
    - F jokaisella muistiviitteellä
    - F laitteistotoimintona? miten?
  - u ei käy, liikaa yleisrasitetta!

Fig 8.15 [Stal05]

# NRU (Not Recently Used) -poistoalgoritmi

## n Sivukehyksissä viitebitti (R, Reference)

- u nollataan aika ajoin
- u laitteisto asettaa, jos viittaus

vaatii siis laitteistotukea!

## n Sivukehyksessä muutettu bitti (M, Modified)

- u nollataan levyllä kirjoituksen yhteydessä
- u laitteisto asettaa, jos kirjoitus

F tarvitaan siis joka tapauksessa!

## n Poista

- u joku, johon ei viittausta ja jota ei muutettu
- u joku, johon ei viittausta ja jota muutettu
- u joku, johon viittaus ja jota ei muutettu
- u joku

## n Helppo toteuttaa, vähän HW-tukea

# FIFO-poistoalgoritmi

- n Korvaa sivu, joka ollut kauimmin muistissa
- n vrt. Round-Robin vuorottamisessa
- n Helppoin toteuttaa
  - u aikaleima sivutaulun alkioon, kun tuodaan muistiin, TAI
  - u kaikki sivut listassa, muistista tuontiajan mukaisesti
- n Muistiintuontiaika korreloi huonosti käyttötarpeen kanssa
  - u kauan muistissa ollut sivu voi olla sellainen, jota käytetään koko ajan?

jonojen hallinta?

Fig 8.15 [Stal05]

# RANDOM poistoalgoritmi

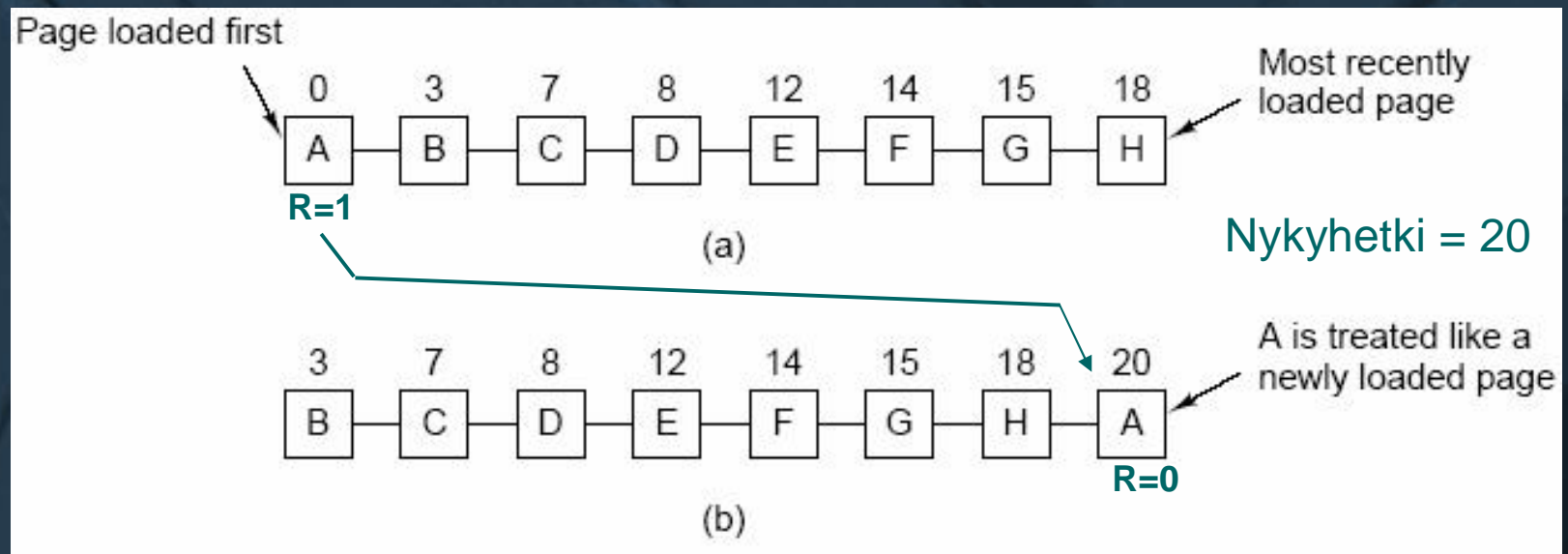
- n Poista satunnainen sivu
- n Todella helppo toteuttaa
- n Ei vaadi laitteistotukea
- n Lähes yhtä hyvä kuin FIFO
  - u ei siis kauhean hyvä...



# Second Chance -poistoalgoritmi

- n FIFO:n modifikaatio
- n viitebitti R ja muutosbitti M
- n poista FIFO-järjestyksessä, mutta
  - u jos  $R=1$  (on viitattu äsken), niin siirrä FIFO jonon loppuun ja nollaa R
  - F jos muita ei löydy, niin sitten tämä poistetaan kun FIFO jonossa päästään seuraavalla kerralla tämän sivun kohdalle (jolloin  $R=0$ )

jonojen hallinta?



(Fig 4-16 [Tane01])



# Clock-poistoalgoritmi

## n Second Chance –algoritmin modifikaatio

- u ei sivujen siirtelyä listassa

## n Käy sivutilojen listaa läpi renkaana

## n Jokaiseen sivutilaan liittyy viitebitti (U eli Use bit)

- u kun sivu muistiin,  $U = 0$ , reset silloin tällöin

- u kun sivuun viitataan,  $U = 1$  (MMU asettaa)

## n Korvaa sivu, jonka viitebitti on 0

- u ensimmäinen eteen sattuva

Fig 8.16 [Stal05]

## n Aseta aina tutkituissa viitebitiksi 0

- u jos arvo seuraavallakin kierroksella 0,  
sivuun “ei ole viitattu” (ainakaan teoriassa)

Fig 8.15 [Stal05]

## n Aseta kaikki $U=0$

- u ei aina? joka 10 ms? (HW-toiminto)

Fig 8.17 [Stal05]

# Clock-poistoalgoritmi M-bitin kera

- n **Lisää laitteistotukea: M eli Modified-bitti**
  - u ennestään U (eli Used eli R eli Referenced)

- n **käy sivuja läpi ringissä**

Not used,  
not modified?

- u jos  $U==0$  ja  $M==0$ , poista tämä
- u älä muuta U-bittiä

- n **käy sivuja läpi uudestaan ringissä**

Not used,  
modified?

- u jos  $U==0$  ja  $M==1$ , poista tämä
- u aseta  $U=0$  muille

- n **käy sivuja läpi uudestaan ringissä**

Not modified?

- u jos löytyy  $M==0$ , niin poista tämä

- n **käy sivuja läpi uudestaan ringissä**

Any page?

- u jos löytyy  $M==1$ , niin poista tämä
- u kaikki jäljellä olevat tällaisia

käy rinki  
läpi monta  
kertaa,  
kunnes  
poistettava  
löytynyt

Fig 8.18 [Stal05]

# Käyttöjärjestelmät II

## Käyttäjoukkostrategia

### Muistitilan koon hallintaan

# Käyttäjöjoukkostrategia (Working Set Strategy)

Fig 8.1 [Stal05]

## n Sivut, joihin viitattu k:n (ikkunakoko) viimeisimmän viittauksen aikana

- u approksimaatio prosessin tarvitsemille sivuille (todelliselle käyttäjöjoukolle)

Fig 8.19 [Stal05]

"working set" vs. "resident set"?

## n Toteutus?

- u vain approksimaatioita
- u käyttäjöjoukossa sivut, joihin viitattu esim. viim. 100 ms:n aikana, kun prosessi oli suoritettavana

## n Sivu, joka ei kuulu käyttäjöjoukkoon, voidaan vapauttaa

- u pyrkii tekemään ennakolta tilaa uusille sivuille

## n Jos käyttäjöjoukko kutistuu pieneksi, voidaan tuoda sivuja ennalta takaisin

- u mitkä? viitattu ja sitä seuraava?

# Käyttäjöjoukon koko

## n Montako sivutilaa siis per prosessi?

- u osa käyttäjöjoukoissa, osa pidettävä vapaana
- u vähän → muistissa monta prosessia, usein puutoksia
- u paljon → vähemmän sivunpuutoskeskeytyksiä
- u paikallisuus vs. sivunpuutos

## n Kiinteä allokointi

- u prosessia käynnistettäessä
- u tyypin perusteella: interaktiivinen, erätyö (tausta)
- u muun tiedon perusteella, esim. edellinen suorituskerta

## n Dynaaminen allokointi

- u sivuja muistiin tarpeen mukaan
- u käyttäjöjoukon koko vaihtelee vapaan muistitilan mukaan ja prosessin vaiheen mukaan

## n Mikä sopiva käyttäjöjoukon koko?

- u arvio esim. sivupuutostiheyden perusteella



# Dynaaminen käyttöjoukon koko

## n PFF – Page Fault Frequency

- u pidetään kirjaa, milloin edellinen sivunpuutos tapahtui
- u sivunpuutos: laske  $T$  = väliaika edellisestä sivunpuutoksesta
- u jos  $T < L$ , niin tarvitaan lisää muistitilaa
  - F lisää puuttuva sivu käyttöjoukkoon
- u jos  $T > H$ , niin prosessilla ”liikaa” muistitilaa
  - F poista kaikki viime aikoina viittaamattomat sivut tai jotkut niistä käyttöjoukosta
- u yritetään pitää sivunpuutostiheys sopivissa rajoissa, eli sivunpuutosten väliaika välillä  $(L, H)$

## n VSWS – Variable-interval Sampled Working Set

- u PFF:n modifikaatio, ottaa huomioon vaiheen vaihtumisen
- u käyttöjoukko voi kasvaa milloin vain, mutta pienentyä vain vaiheen vaihtuessa
- u vaiheen pituus aina vähintään  $M$  ja enintään  $L$
- u vaiheen aikana hyväksyttävän sivunpuutosmäärän  $(Q)$  avulla



# Käyttöjärjestelmät II

## Levyille kirjoitus –politiikka (Cleaning Policy)

*”how to clean dirty pages”*

# Milloin sivu kirjoitetaan levyille?

n **Vain jos muutettu**

n **Tarvetalletus** (demand cleaning)

- u kirjoita levyille vasta, jos sivutila pitää ottaa käyttöön
- u saattaa aiheuttaa pitkän viipeen sivutilan tarvitsijalle
- u entä jos kone kaatuu?

n **Ennaltatalletus** (precleaning)

- u kirjoitus levyille etukäteen erä kerrallaan
- u ennaltamääritellyin aikavälein

n **Sivutila siivouksen jälkeen vapaiden listaan**

- u siivous voi olla hukka-investointi
- u entäpä, jos sivuun viitataan taas?

n **Entäpä jos levy odottaa toimetonna, eli sillä on pieni utilisaatio?**

# Puskurointi (page buffering)

n Pidä tietty osa sivutiloista vapaana

- u joskus tulee vapautetuksi väärä sivu

~ sivujen välimuisti

- u nopea palautus takaisin käyttöön sivupuskurista, jos "vapaassa" sivukehyksessä olevaan tietoon viitataan

n Poistettavaksi merkitty sivu lisätään

- u vapaiden listaan, jos ei muutettu
- u muutettujen sivujen listaan, jos muutettu

n Sivupysyvyys alkuperäisellä paikalla muistissa

- u vain merkintä prosessin sivutaulussa poistetaan

n Varaus vapaiden listan alusta

- u viimeksi vapautetuilla suurempi mahdollisuus tulla "pelastetuksi"  
F miksi?

# Käyttöjärjestelmät II

## Moniajoaste

# Moniajoaste (load control)

n Löydettävä sopiva suorituksessa (muistissa) olevien prosessien lkm

n Liian vähän?

u CPU jouten, jos prosessit Blocked tilassa

Fig 8.11 [Stal05]

n Liian paljon?

u ruuhkautuminen (trashing): sivuttaminen vie liikaa aikaa

Fig 8.21 [Stal05]

n Mitä tehdä jos liikaa prosesseja?

Eli siis liian vähän muistia per prosessi?

u heittovaihta Ready-prosesseja levyille

F se, jolla pienin prioriteetti

F se, joka aiheutti puutoksen

F se, joka oli edellisenä suorituksessa

F se, jolla pienin käyttäjoukko

F se, jolla eniten muistia käytössä tai jolla suurin aikaviipale

F kuka vaan, mutta joku!

# Moniajoasteen dynaaminen kontrolli

## n L=S –kriteeri

- u sivunpuutosten väliaika L
- u sivunpuutosten käsittelyaika S
- u  $L \gg S \rightarrow$  kasvata mpl,  $L \ll S \rightarrow$  pienennä mpl

## n 50% sääntö

- u Util (sivutuslevy) olisi hyvä olla 50%
- u "pieni" U  $\rightarrow$  kasvata mpl, "iso" U  $\rightarrow$  pienennä mpl

## n Globaali Clock poistoalgoritmi, viisarin nopeusrajat

- u "hidas"  $\rightarrow$  kasvata mpl
- u "nopea"  $\rightarrow$  pienennä mpl



# Käyttöjärjestelmät II

## *Heittovaihtovalue*

# Heittovaihtoalue (swap area, VM backup store)

## n Heittovaihtoalue varattu levytä etukäteen

- u Windows: pagefile.sys, win386.swp
- u Linux: oma swap-levyartitio

## n a) Kiinteä yhtenäinen varaus

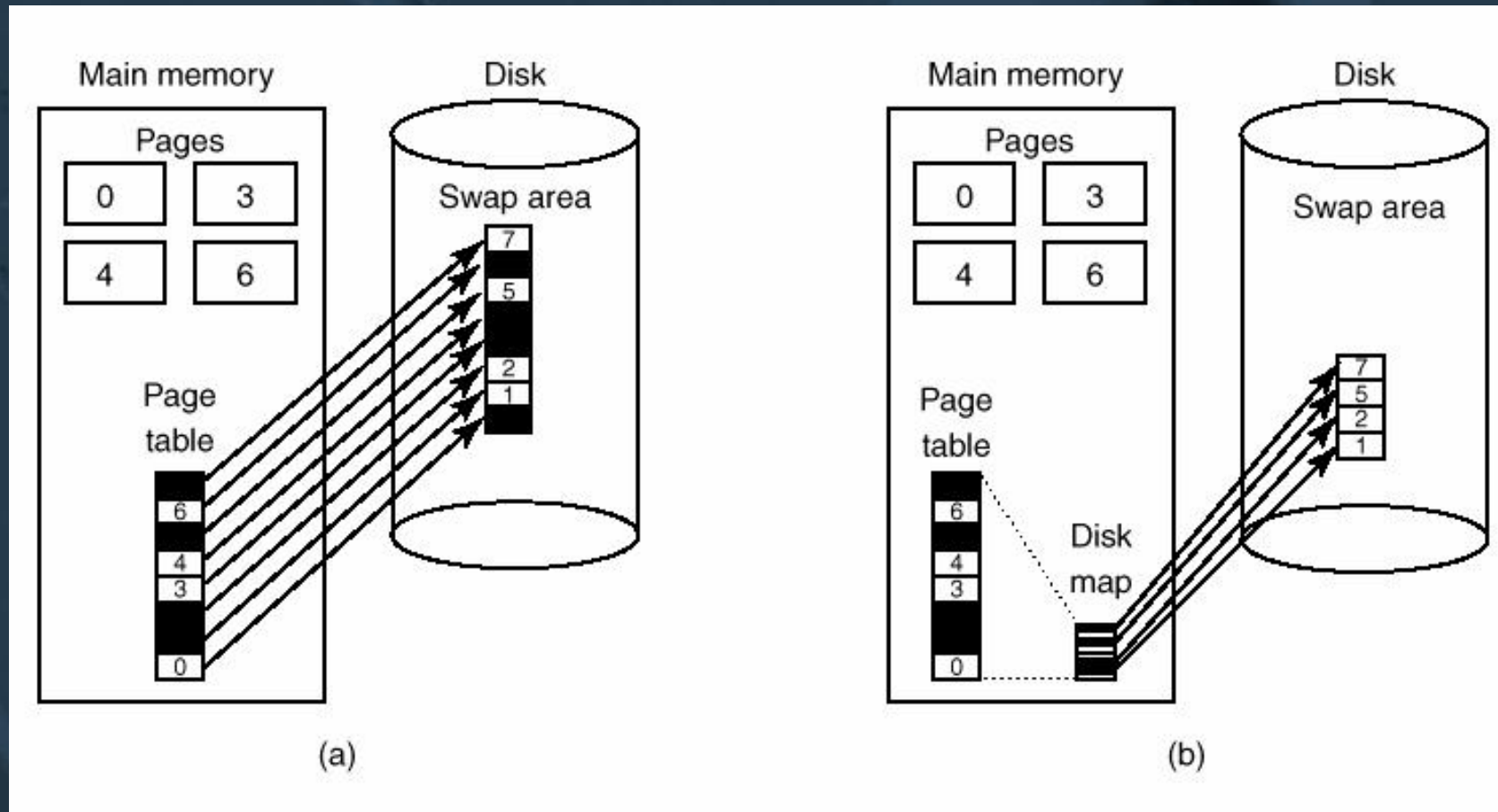
- u kerralla koko prosessille
  - F kopioi koodi alustuksessa tai
  - F varaa tila, heittovaihda sivut sinne vähitellen
- u PCB:ssa swap-alueen alkuosoite ja pituus
- u vapaiden alueiden kirjanpito kuten luvussa 7

## n b) Dynaaminen varaus

- u sivu kerrallaan
- u tarvitaan sivukohtainen kuvaus
  - F sivutaulun alkiossa swap-alueen lohkonnumero tai
  - F erillinen taulu (disk map)
- u ei tarvita levytilaa sivuille, joita ei koskaan viedä levyille

# Heittovaihtovalue

(Fig 4-33 [Tane01])



Kiinteä varaus

Dynaaminen varaus

# Kertauskysymyksiä

- n Miksi PFF:ssä kannattaa olla kaksi rajaa eikä vain yksi?
- n Miten OPT voidaan käytännössä toteuttaa?
- n Mikä on OPT:n merkitys?
- n Mitä hyötyä kotikoneessa on suuresta moniajoasteesta?
- n Miksi Clock-algoritmi asettaa ohitettaville sivuille  $use=0$ ?
- n Miksei sijoituspolitiikka ole niin kiinnostava?
- n Mitkä hyötyä on vapaiden sivujen puskuroidinnista?  
Miten sen aiheuttamia haittoja voidaan minimoida?